Runtime Assurance for Big Data Systems

John Klein

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software Engineering Institute | **Carnegie Mellon University**

SEI Research Review 2015
October 7–8, 2015
© 2015 Carnegie Mellon University
Distribution Statement A: Approved for Public Release;
Distribution is Unlimited

2

System Measurement for Assurance at Runtime

Big Data systems have a very dynamic runtime context – assurance "by design" is still necessary but not sufficient

- New and changing data sources – sensors, humans, systems
- Evolving user workloads driven by new missions and new data
- Shared infrastructure – variable quality of service

Need to monitor the big data system in its runtime environment

- Collect measurements/metrics
- Assess health and trigger action to assure capability delivery

Software Engineering Institute | **Carnegie Mellon University**

3

System Measurement for Assurance at Runtime

Big Data systems have a very dynamic runtime context – assurance "by design" is still necessary but not sufficient

- New and changing data sources – sensors,
- Evolving user workloads driven by new miss
- Shared infrastructure – variable quality of ser

Focus of this year's project

Need to monitor the big data system in its runtime environment

- Collect measurements/metrics
- Assess health and trigger action to assure capability delivery

Future Work – Predict, Diagnose

Software Engineering Institute | **Carnegie Mellon University**

SEI Research Review 2015
October 7–8, 2015

© 2015 Carnegie Mellon University
Distribution Statement A: Approved for Public Release;
Distribution is Unlimited

4

Technical Challenges – Measurement Collection in Big Data Systems

System scale

- 1000s of nodes
- Millions of measurement time series streams
- Efficiency is critical

Processing and storage framework resiliency and redundancy makes individual node status less meaningful

- Need aggregate application-level measurements composed from component data
- End-to-end system performance is the ultimate health measure

Software Engineering Institute | **Carnegie Mellon University**

Solution Approach

Scale → Automation

- Monitor generation

- Monitor insertion

- Measurement collection and aggregation

- (Future) Generate visualizations

Architecture Styles → Abstractions and constraints to enable efficient automation (metamodels)

- Styles/Patterns capture common architecture approaches

  - Restrict types of components and topologies

  - Establish semantics for a class of functionality

  - Define what to measure, where to measure, how to aggregate

6

Solution Approach

Scale → Automation

- Monitor generation
- Monitor insertion
- Measurement collection and aggregation
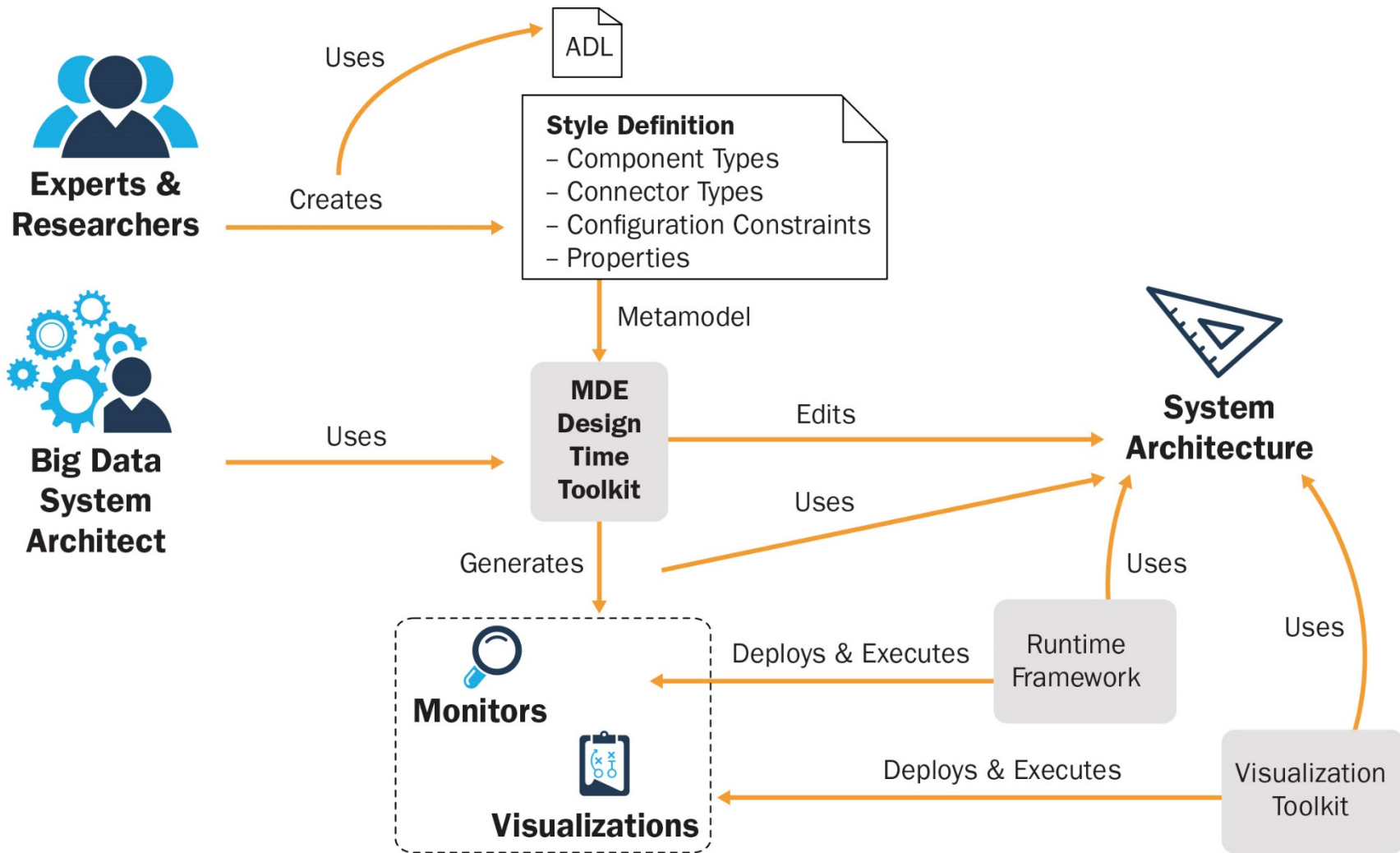- (Future) Generate visualizations

Architecture Styles → Abstractions and constraints to enable efficient automation (metamodels)

- Styles/Patterns capture common architecture approaches
  - Restrict types of components and topologies
  - Establish semantics for a class of functionality
 - Define what to measure, where to measure, how to aggregate

New Contribution

Software Engineering Institute | **Carnegie Mellon University**

SEI Research Review 2015
October 7–8, 2015

© 2015 Carnegie Mellon University
Distribution Statement A: Approved for Public Release;
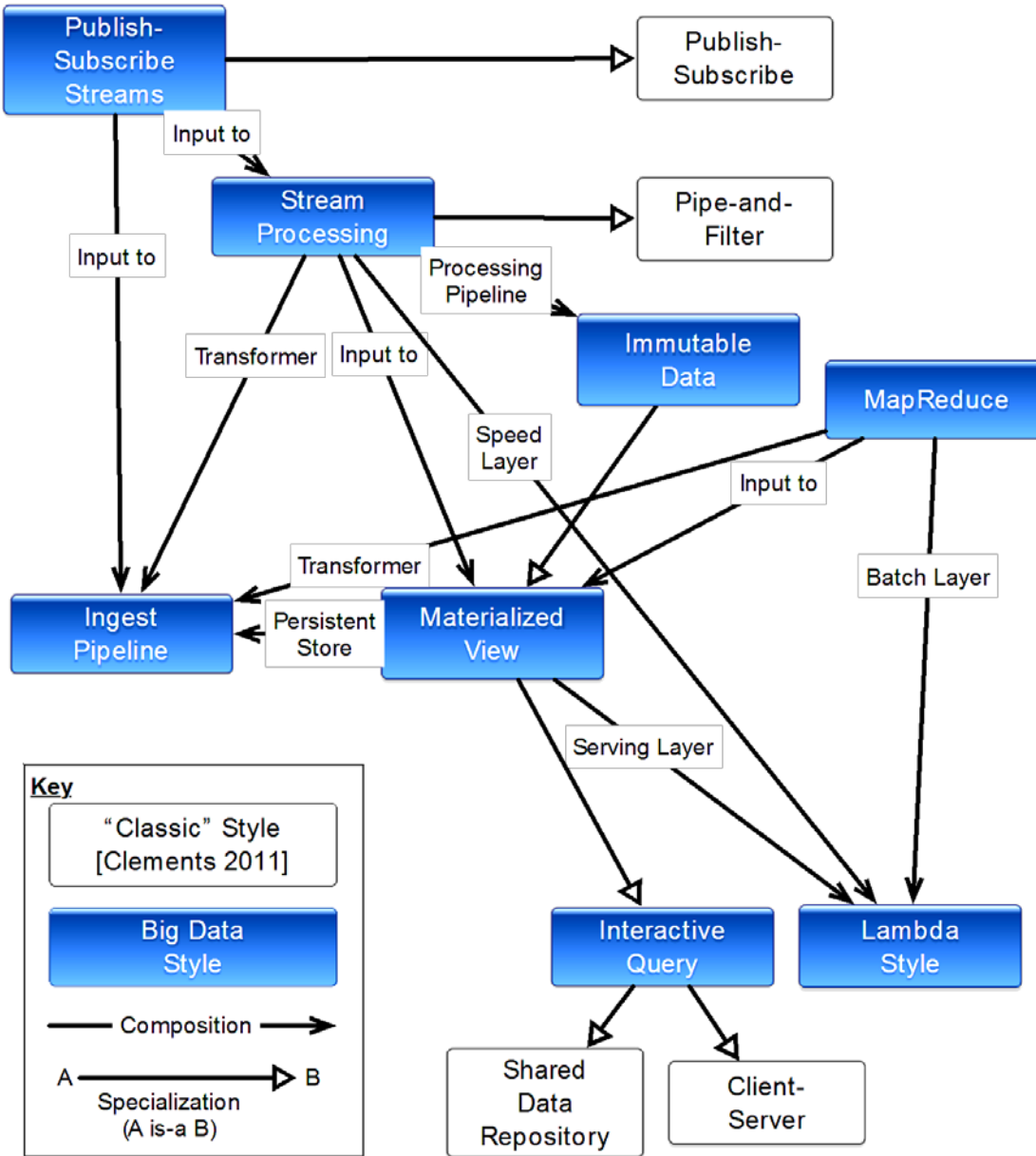Distribution is Unlimited

7

Scale → Automation
Leverage Architecture Styles to Automate

Results –
Architecture Styles for Big Data Systems
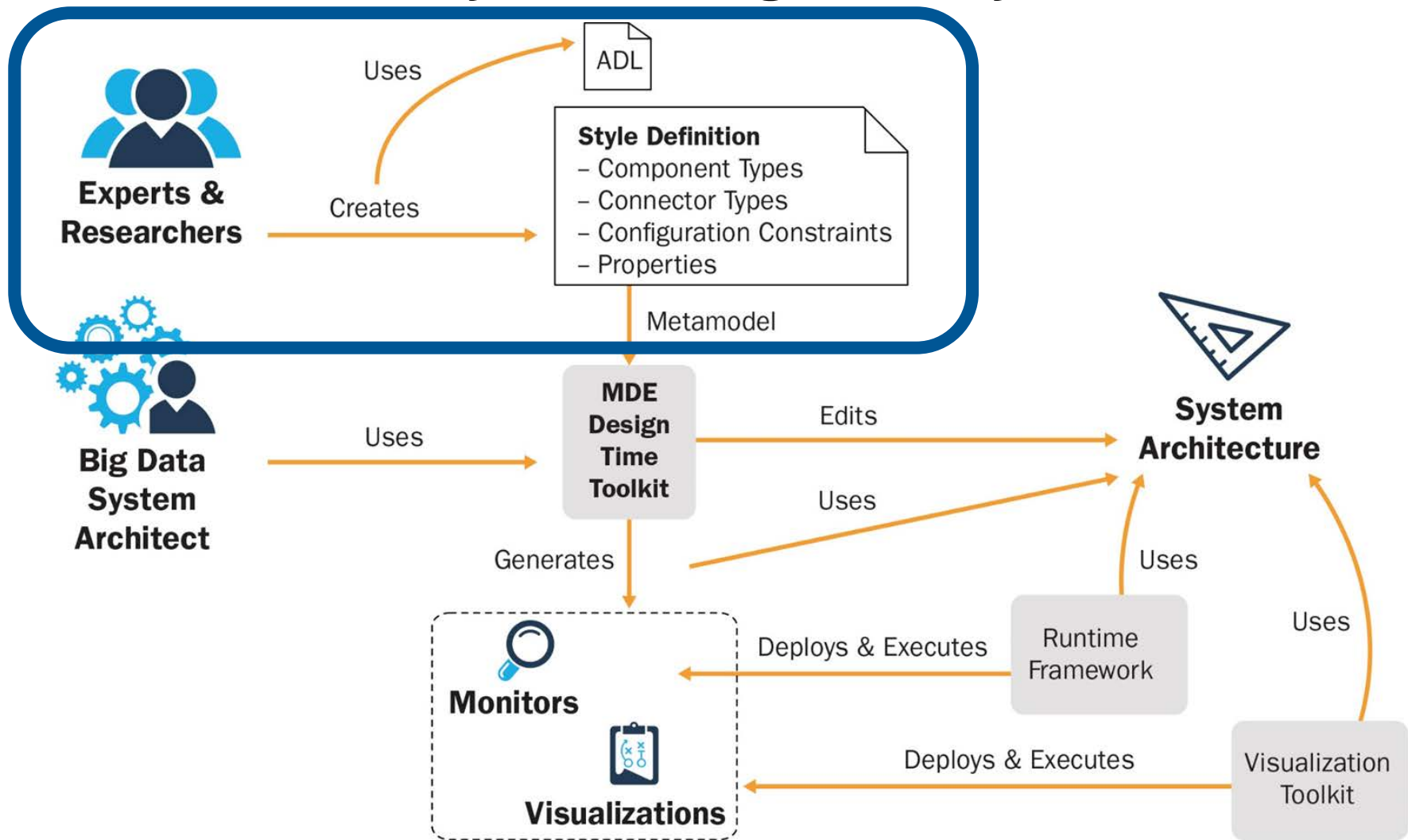
Architecture Styles for Data-Intensive Systems

Example – Ingest Pipeline Style
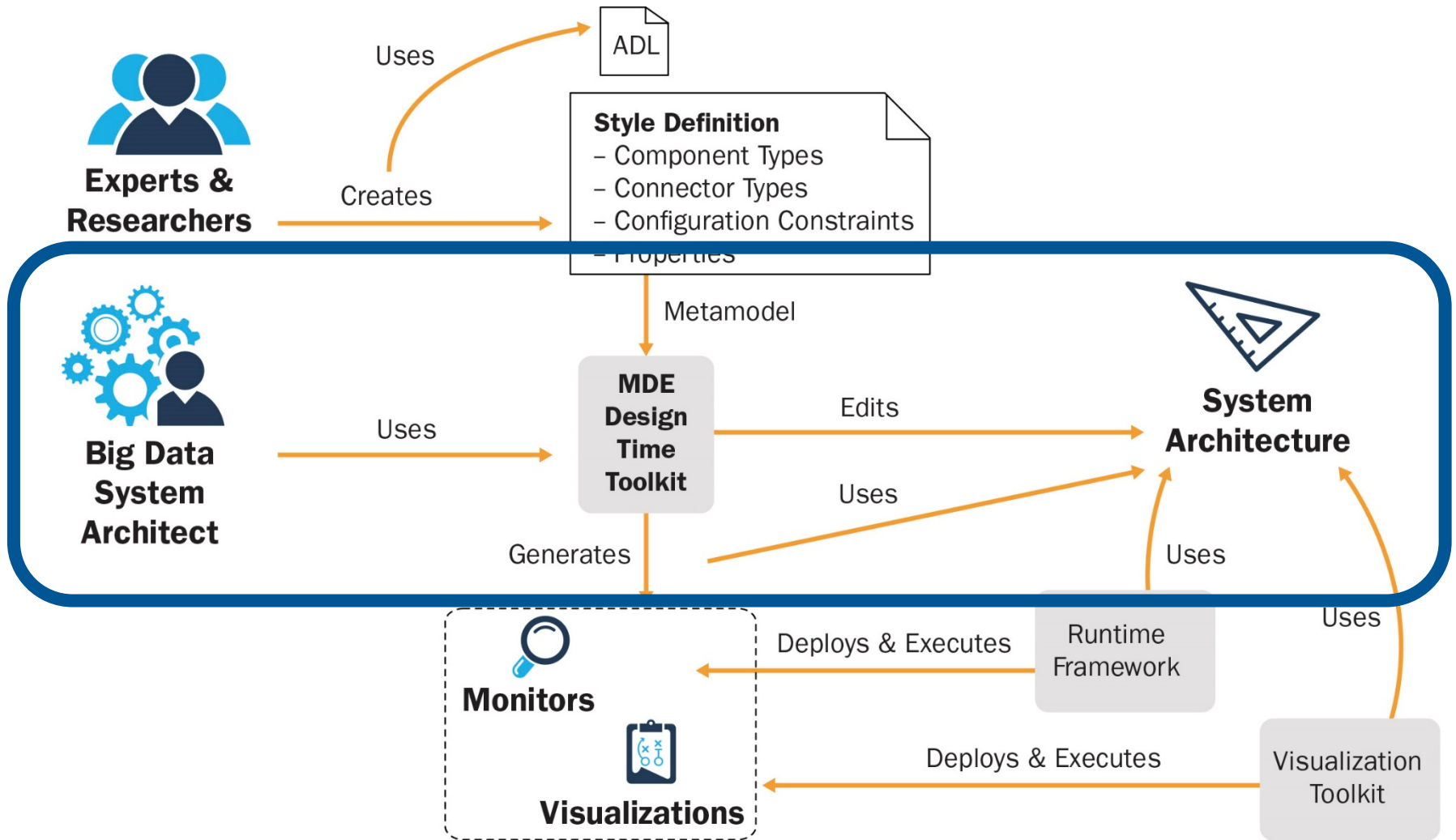
Component Types

- Source Adapter, Transformer, Writer, Persistent Store

Measurement Metamodel

- A healthy instance of the _ingest pipeline style_ processes data at a rate that keeps ahead of incoming data. The measurements to provide visibility into this include:

  - # of input messages/records and rate for each input source

  - # of data store writes per namespace and write rate

  - Application-specific counts of values of particular input and output data types ("peg counts" or histograms) to assess the distribution of input and output data sets.

Software Engineering Institute | **Carnegie Mellon University**

Results –
Using Styles to Model Big Data Systems

Software Engineering Institute | Carnegie Mellon University

12

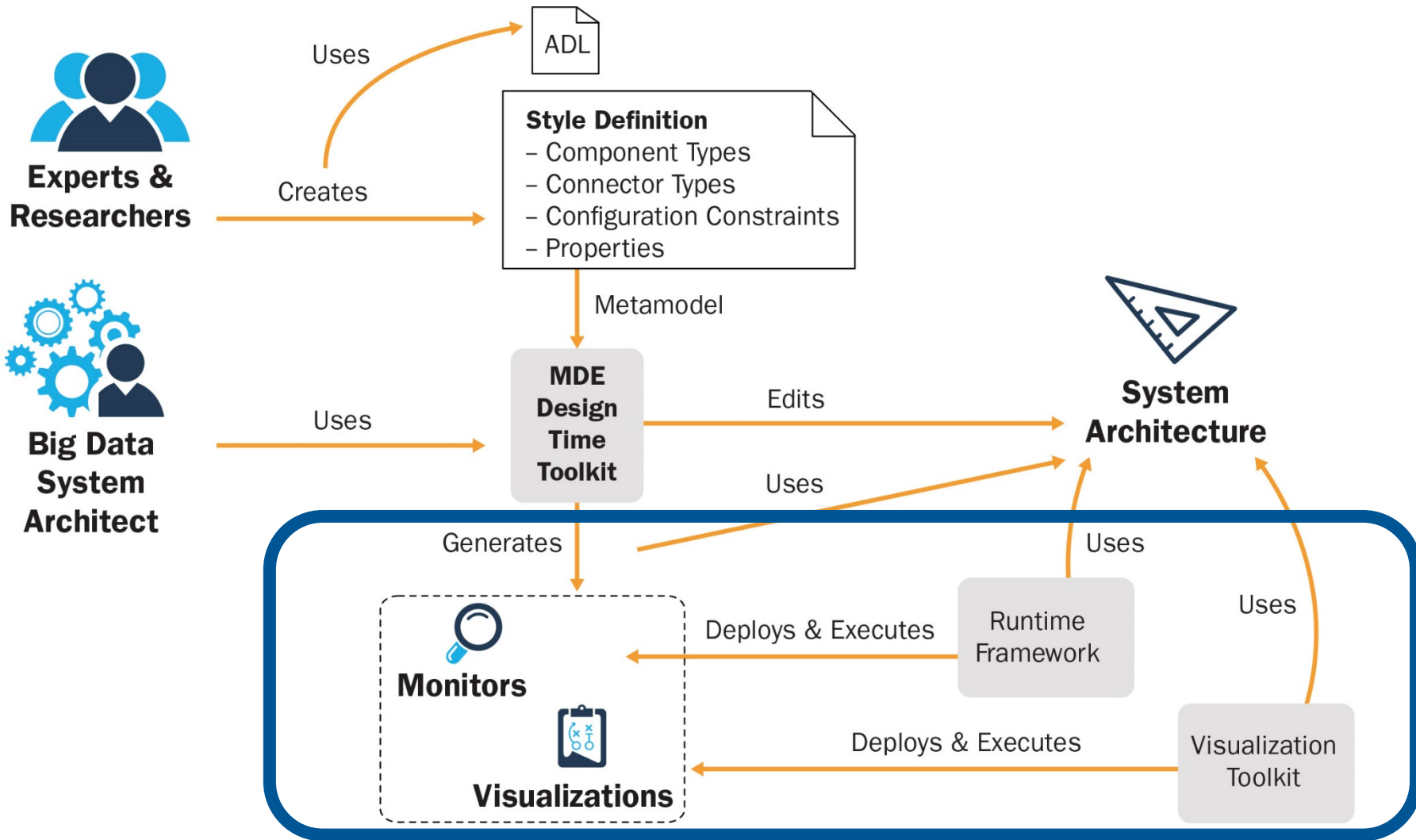# Style-Based System Modeling in Acme Architecture Description Language
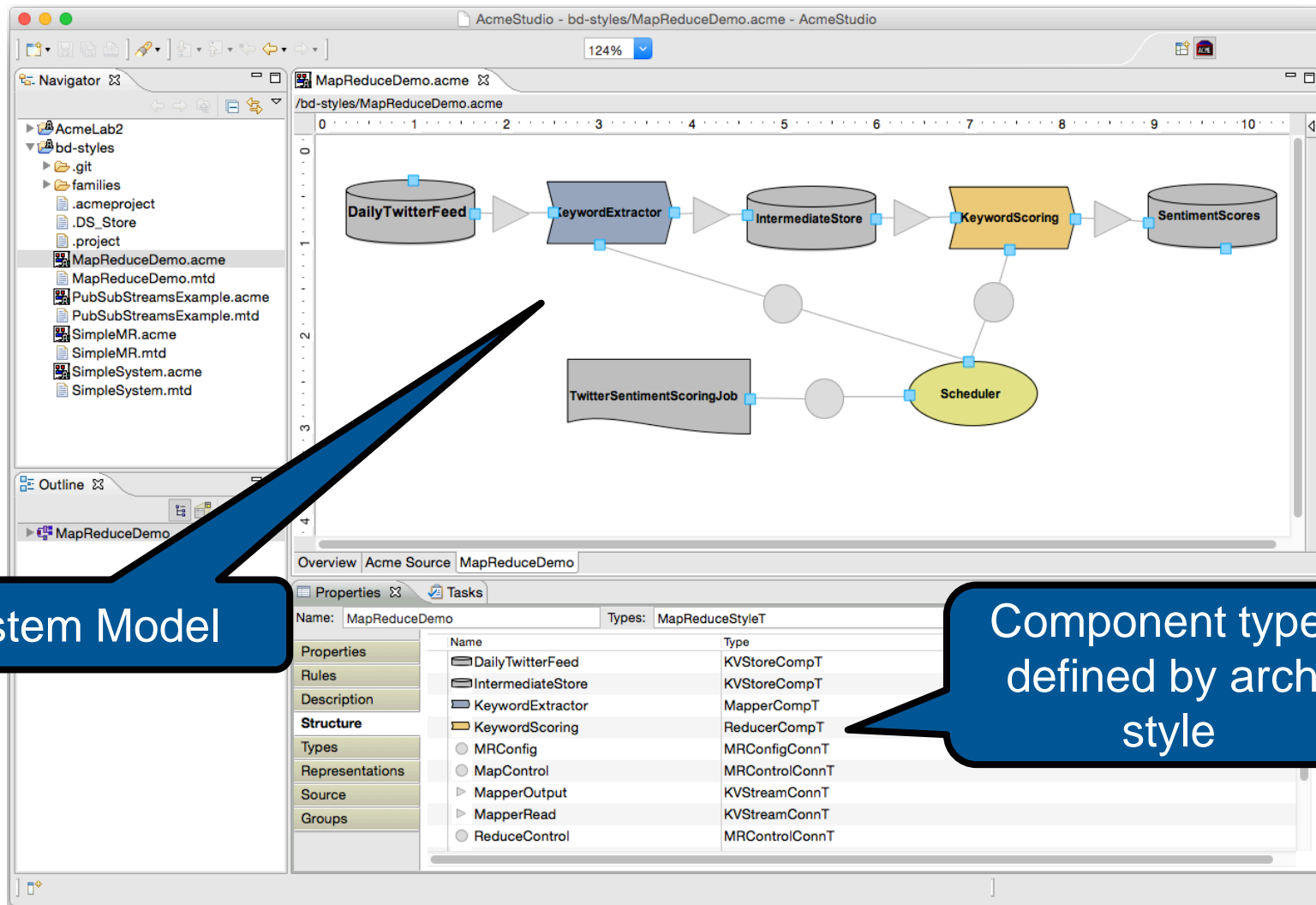
System Model

Component types defined by arch. style

Software Engineering Institute | Carnegie Mellon University

Results – Automated Monitor Generation and Measurement Collection

Model-Generated Monitors and Collection for NoSQL Persistence (CMU MSE Project)

Multiple NoSQL Databases

Model-Driven

Software Engineering Institute | **Carnegie Mellon University**
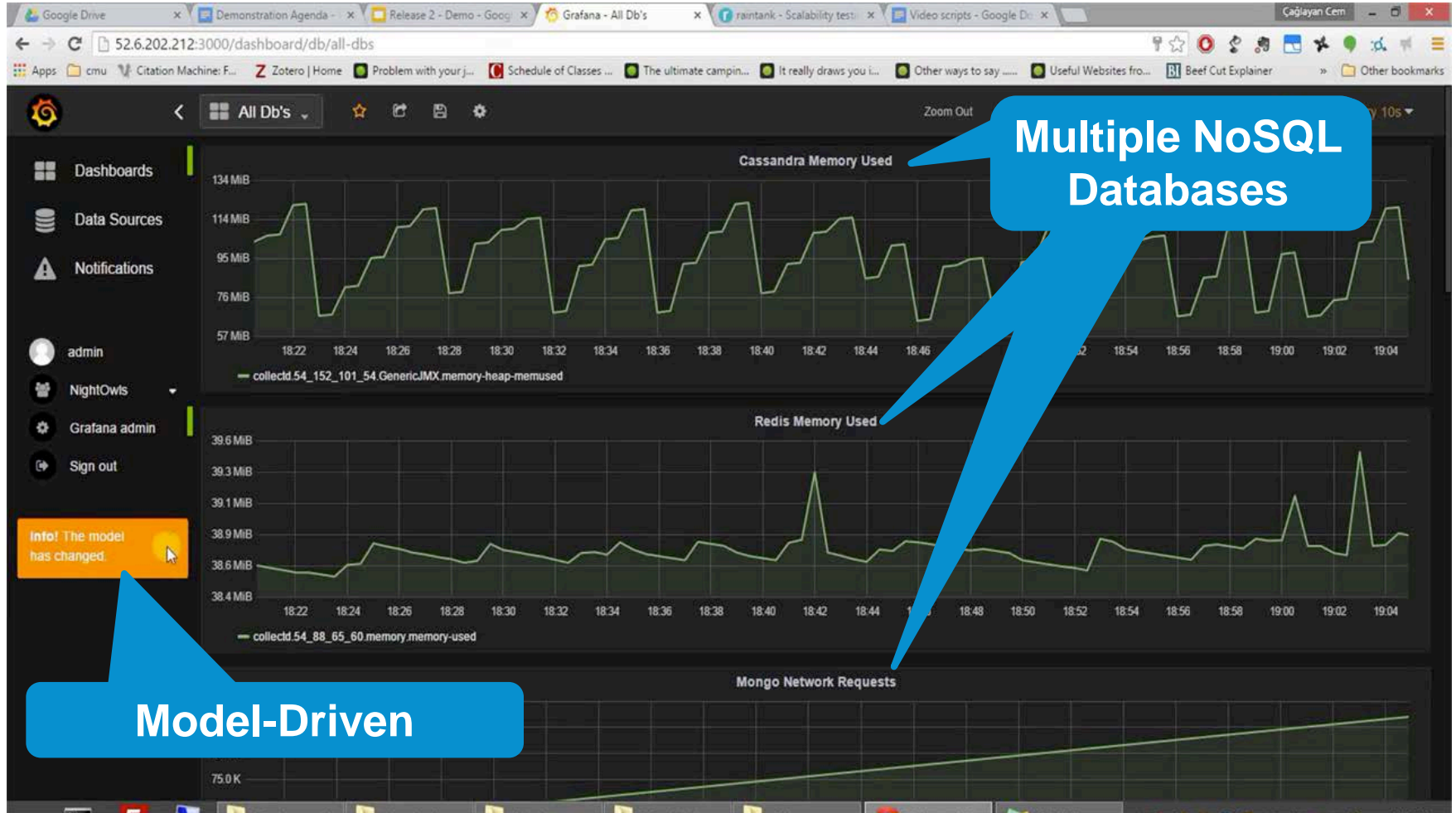
Results and Future Work

Demonstrated feasibility of automating measurement collection and aggregation

- Steps are manually integrated
- Limited monitor generation

Future Work:

- Transparent monitor insertion into existing systems that use open source processing frameworks (e.g., NTC using Apache Storm)
- Extension of style catalog for machine learning systems
- Automated generation of advanced visualizations
- Anomaly detection analytics (e.g., analytics like Netflix "starts per second" analytics)

Software Engineering Institute | **Carnegie Mellon University**

Contact Information

John Klein

Senior Member of Technical Staff

Software Solutions Division

Telephone: +1 617-283-2170

Email: jklein@sei.cmu.edu

Web

www.sei.cmu.edu

www.sei.cmu.edu/contact.cfm

U.S. Mail

Software Engineering Institute

Customer Relations

4500 Fifth Avenue

Pittsburgh, PA 15213-2612

USA

Customer Relations

Email: info@sei.cmu.edu

Telephone: +1 412-268-5800

SEI Phone: +1 412-268-5800

SEI Fax: +1 412-268-6257